EA : Cooperative Inverse Reinforcement Learning

Link to the associated code

Martin Drieux, Adrien Goldszal, Edouard Rabasse

Contents

1 Introduction

In reinforcement learning (RL), an agent interacts with its environment, maximizing its cumulative reward over time and therefore learning a policy dictating its actions based on states and observations. However, one critical issue in RL is the alignment problem: if the reward function is not correct, the result computed will not be correct either. The Inverse Reinforcement Learning (IRL) framework tries to address this by estimating reward functions by observing behaviors that maximize that reward function (for instance, a recommendation algorithm that infers your tastes from the videos you watch). The Cooperative Inverse Reinforcement Learning (CIRL) framework goes one step further: it introduces an agent which has perfect knowledge of the reward function (referred to as the "human" in the article), which is incentivized to convey it to another agent with its behavior (the other agent is called a "robot"). Although this work was initially developed for robotics, the terms "human" and "robot" can refer to a much broader class of agents.

2 Paper Presentation

2.1 The CIRL Framework

Preliminary Assumptions From the start, the paper makes some important assumptions on the CIRL framework and applies it to a specific human and robot collaboration scenario, moving away from the general CIRL. More specifically, in this context, the human (H) knows the complete reward function.

Knowing these specificities, we can formally define the CIRL framework in our case as :

$$M = \langle X, \{\mathcal{A}^H, \mathcal{A}^R\}, T(\cdot \mid \cdot, \cdot, \cdot), \{\Theta, R(\cdot, \cdot, \cdot; \cdot)\}, P_0(\cdot, \cdot), \gamma, h \rangle$$

where:

- X: State space.
- $\mathcal{A}^H$: Human action space.
- $\mathcal{A}^R$: Robotic action space.
- $T(s' \mid s, a_H, a_R)$: Transition dynamics : probability of going from a state s to state s', after actions a_H and a_R
- Θ: Reward parameter space (it is used to define the reward function)
- $R(\cdot, \cdot, \cdot; \cdot)$: Reward function (depends on current states actions, and is parametrized by θ)
- $P_0(\cdot, \cdot)$: Initial and reward parameter and state distribution.
- γ: Discount factor.
- $h \in \mathbb{R}$: Horizon of the problem (number of states)

2.2 Key Principles

- Solving the CIRL means finding joint optimal policies (which are functions that map the history of actions and states to actions for the next step)
- The reward function is shared for both agents, which incentivizes collaboration. That makes it differ from RL and IRL
- The human may act sub-optimally (by comparing to a situation where he would only maximize its own function) to convey information about Θ. -
- The robot interprets the human's actions to update its belief about Θ.

2.3 From CIRL to POMDP

2.3.1 POMDP definition

A partially observed decision process (PODMP) extends a MDP and is defined as follows : $M_{POMDP} = \langle X, \mathcal{A}, Z_{observation}, O(\cdot \mid \cdot, \cdot), T(\cdot \mid \cdot, \cdot), \{\Theta, R(\cdot, \cdot; \cdot)\}, P_0(\cdot, \cdot), \gamma, h\rangle$

- S : State space
- A : Set of actions
- $T(x'|x, a)$: Transitions, a probability distribution describing how the environment transitions from states x to x' given action a
- $Z_{observation}$: observation space, partial information the agent receives about the state
- $O(o|x', a)$: observation function, probability distribution over observations given the new state s' and action a.
- $R(x)$: reward function the agent receives in state x
- b : belief state, a probability distribution over states that represent the agent estimate of the current state.
- $P_0(\cdot, \cdot)$: Initial state distribution.
- γ: Discount factor.
- $h \in \mathbb{R}$: Learning phase duration.

2.3.2 Reformulation of CIRL as POMDP

Initially, we might see the CIRL as a multi-agent POMDP, computing the optimal joint policy for this kind of problem is NEXP-complete. However, the paper proves that CIRL problem can be reduced to a *coordination*-POMDP (which is still expensive to compute but more reasonable, see section 4.3 for more details). In this setup, the single actor is a coordinator C. The states are $S_C = S \times \Theta$, where S was the original state space. C actions are (δ^H, a^R), specifying an action for the robot R and a decision rule for the human H : $\delta^H : \Theta \to A^H$, depending on the reward parameter.

The coordinator observes the human actions and updates its belief $b(\theta)$ on the distribution of θ. We can describe the CIRL as a coordPOMDP as follows :

$$M' = \langle (X \times \Theta), (A^{H^\Theta} \times A^R), (X \times A^H), O, T', \Theta, R(.|.), \gamma, P_O \rangle$$

The transition T' is defined as follows :

$$T'((x', \theta')|(x, \theta), (a_R, \delta^H) = \mathbb{1}_{\theta=\theta'} T(x'|x, a_R, \delta^H(\theta))$$

where T is the original transition dynamic.

The observation function O is :

$$O(x', a^{H'}|x, \delta^H, a^R, \theta) = \mathbb{1}(x', a^{H'}) = (x, \delta^H(\theta)))$$

The coordinator only observes the physical state and the human action.

The *coordination*-POMDP is illustrated on figure 1.

Then, the paper uses a property of POMDP [3] : an optimal policy only depends on the belief state. As a result, we can compute the best policy for both the human and the robot knowing only $b(\theta)$ and without knowing the history of actions.

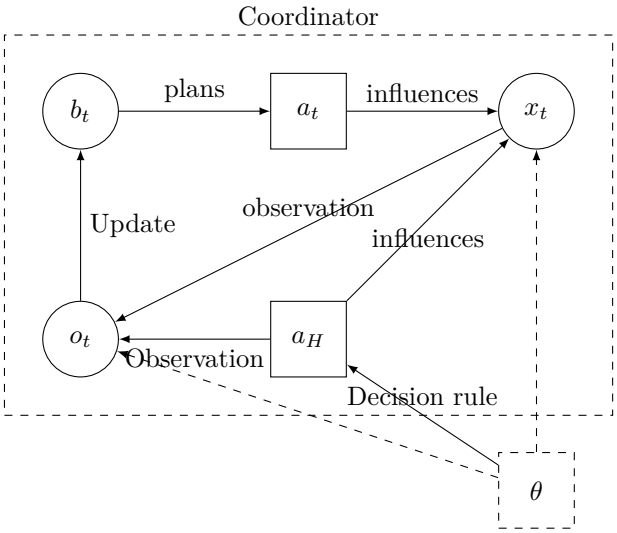

Figure 1: Coordination-POMDP

2.4 Clarification of the ACIRL

The authors introduce Apprenticeship CIRL a turn-based game composed of 2 phases : a learning phase during which only the human takes actions, and a deployment phase where only the robot acts. The author

claim that this a type of CIRL, which is not obvious at first since only one actor takes actions during each phase. However, enforcing strong constraints on the policies, we managed to formulate this as a special type of CIRL. The ACIRL is defined as follows:

$$M = \langle S, \{\mathcal{A}^{H^\Theta}, \mathcal{A}^R\}, T(\cdot \mid \cdot, \cdot, \cdot), \{\Theta, R(\cdot, \cdot, \cdot; \cdot)\}, P_0(\cdot, \cdot), \gamma, h \rangle$$

where:

- S: state space,
- $\mathcal{A}^H$: human action space,
- $\mathcal{A}^R$: robotic action space,
- $T(\cdot \mid \cdot, \cdot, \cdot)$: transition dynamics,
- Θ: reward parameter space,
- $R(\cdot, \cdot, \cdot; \cdot)$: reward function,
- $P_0(\cdot, \cdot)$: initial state distribution,
- γ: discount factor,
- $h \in \mathbb{R}$: learning phase duration.

Under the additional constraint that there exists a NOOP (no operation) action such that:

$\text{NOOP} \in \mathcal{A}^R, \text{NOOP} \in \mathcal{A}^H$ (no operation) such that:

And the admissible policies are such that (the * operator designates an history):

$$\forall u^H \in [\mathcal{A}^H \times \mathcal{A}^R \times S]^* \times \Theta, \quad \pi^H(u^H) = \text{NOOP} \quad \text{if length of } u^H > h$$

$$\forall u^R \in [\mathcal{A}^H \times \mathcal{A}^R \times S]^*, \quad \pi^R(u^R) = \text{NOOP} \quad \text{if length of } u^R \leq h$$

In this formulation:

- u^H represents the history of human and robotic actions as well as states during the learning phase.
- u^R represents the history of robotic actions, human actions, and states.
- h is used to separate the learning and deployment phases.
- The human policy π^H becomes inactive (NOOP) after the learning phase (length of $u^H > h$).
- The robotic policy π^R remains inactive (NOOP) as long as the learning phase is not completed (length of $u^R \leq h$).

2.5 Generating instructive demonstrations

In this ACIRL environment, the robot's optimal policy greedily maximizes reward from the mean θ from its belief in the deployment phase :

$$\pi_R^* = \arg\max_{\pi^R} \mathbb{E}\left[R(s, \pi_R(s) \mid \theta)\right]$$

The key insight here is that, knowing this robot behaviour, the human's best response may not a demonstration by expert (DBE) maximizing its own reward.

To establish some results on ACIRL and possible best responses, the paper uses a specific framework and makes some assumptions without stating them explicitly. Here are these assumptions:

1. Rewards are a linear combination of state features for a certain feature function :

   $$\phi : R(s, a^{\mathcal{H}}, a^{\mathcal{R}}; \theta) = \phi(s)^\top \theta$$

   In that case, two policies with the same expected feature counts μ will have the same values.

   $$\begin{aligned} V^\pi &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t \mid \pi)\right] \\ &= \theta \cdot \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t) \mid \pi\right] \\ &= \theta \cdot \mu(\pi) \end{aligned}$$

   This means that a learned policy that matches the feature counts of the observed policy will be as good as the demonstration for any given θ

2. **R** uses an IRL algorithm that computes its belief on θ **specifically by maximizing feature counts**, as Maximum-Entropy IRL for example.

This two assumptions ensure that the robot's policy

$$\pi_R^* = \arg\max E_\pi[\Phi(\pi)] \cdot \theta$$

will match the expected feature counts of the learning phase. In fact, we see through this equation that there is no reason the expected feature counts match if the belief through IRL wasn't computed through an algorithm that does. This underlying assumption wasn't detailed in the paper. In fact, there exist IRL algorithms such as Adversarial IRL that do not compute a belief on θ by matching expected feature counts.

Knowing this feature count matching, the paper proposes the following heuristic for the human, to generate better demonstrations by matching the feature counts with those of the *distribution of trajectories* induced by θ : ϕ_θ

$$\tau^H \leftarrow \arg\max_{\tau} \underbrace{\phi(\tau)\cdot\theta}_{\substack{\text{Sum of rewards}\\ \text{for the human}}} \underbrace{-\eta\|\phi_\theta - \phi(\tau)\|^2}_{\substack{\text{Feature}\\ \text{dissimilarity}}}$$

3 Our implementation of the paper

3.1 Implementation details

We implement this turn-based ACIRL on a 10x10 gridworld environment. The paper provided the heuristic as well as the overall method; however, no code was provided, and concrete implementation choices were ours.

In this setting, n feature centers are chosen on the grid. The vector $\phi(x)$ of size n represents the distance of x to each of those centers. For each state, the agents obtain a reward $\phi(x)^T\theta$. Only the human knows θ, while the robot knows where the feature centers are but not the values associated with them.

The goal for the agents is to maximize the global reward: $\sum_{t=1}^{T}\gamma^t\phi(x_t)\cdot\theta$.

3.1.1 Demonstration by Expert Policy

The demonstration by expert greedily maximizes reward on its turn. To get this DBE policy, we first calculate the value function associated with the policy by iteratively updating the values for each state until convergence through a Bellman update:

$$V(s) \leftarrow \max_a[R(s,a,s') + \gamma V(s')]$$

We can then get the policy from the value function through the same process:

$$\pi^*_{DBE}(s) = \arg\max_a[R(s,a,s') + \gamma V(s')]$$

3.1.2 Best Response

To get a working best response, the expected feature counts of the whole distribution of trajectories induced by the real θ need to be calculated: ϕ_θ. We calculate this value by averaging over the feature counts of the policies from the optimal trajectory starting from each possible state on the grid.

3.1.3 Maximum Entropy IRL

Maximum Entropy IRL[4] is a type of IRL that is based on feature expectation matching. The idea is to find a distribution of trajectories that maximizes entropy (and therefore minimizes information, i.e., any biases), as there are multiple solutions to the optimization problem.

We find the probability distribution over trajectories $p(\tau)$, or, knowing our reward structure $p(\tau \mid \theta)$, maximizing entropy **H** under constraints:

$$\arg\max_p H(p)$$
$$\text{subject to} \quad \mathbb{E}_{\pi^H}[\phi(\tau)] = \mathbb{E}_{\pi^R}[\phi(\tau)]$$
$$\sum_\tau p(\tau) = 1, \quad \forall\tau : p(\tau) \geq 0 \quad \text{(probability constraints)}$$

We get $p(\tau \mid \theta)$ by solving our optimization problem through Lagrange multipliers, and then optimize for θ by maximizing the log-likelihood.

$$\theta^* = \arg\max_\theta \mathcal{L}(\theta) = \arg\max_\theta \log p(\tau \mid \theta)$$

Deriving, we get the gradient:

$$\nabla_\theta\mathcal{L}(\theta) = \mathbb{E}_{\pi^E}[\phi(\tau)] - p(\tau \mid \theta)\phi(\tau) = \mathbb{E}_{\pi^E}[\phi(\tau)] - D_\tau\phi(s_\tau)$$

where D_τ is the state visitation frequency.

How does the algorithm work? We adapted the implementation from Maximilian Luz[1], PhD student at the University of Freiburg.

Knowing the terminal state of the trajectory, we can compute through a **backward pass** the **local action probabilities**, i.e., the probabilities of selecting a particular action in a given state, as inferred by the demonstration trajectory. We can then calculate (**forward pass**) the **state visitation frequencies (SVF)**, knowing the initial position of the demonstration trajectory.

3.2 Result analysis and comparison

We ran the ACIRL framework on the 10x10 grid, with a horizon of 16 (trajectory length of 8 for human and robot), for 3 and 10 feature counts. This simulation was run 50 times and the results analyzed.

3.2.1 Trajectories

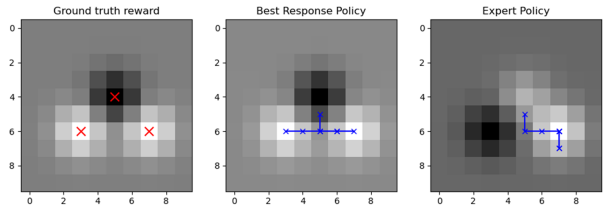

Figure 2: Best response vs Expert demonstration with inferred reward values from the robot

Figure 1 illustrates the impact of the feature count matching term in the best response in this example with three features. The best response trajectory explicitly goes to the two lower feature centers instead of just maximizing its own reward (DBE). The inferred θ by the robot and therefore the reward structure are closer to the ground truth in the best response thanks to the better demonstration.

3.2.2 Results

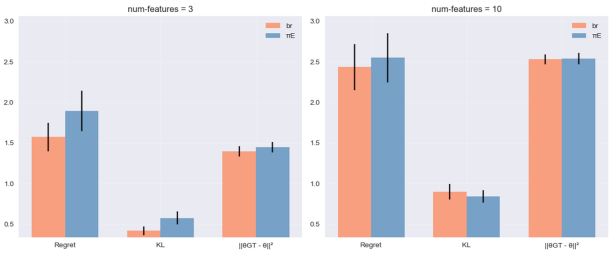

Figure 3: Results comparison: Best response vs Expert demonstration

We compare the results of the best response and expert trajectories on three similarity measures:

- **Regret**: the difference in the value of the policy that maximizes the inferred $\hat{\theta}$ and the value of the policy that maximizes the true θ.

$$regret = V^{\pi_{\hat{\theta}}} - V^{\pi_\theta}$$

- The **KL divergence** between the trajectory distributions induced by θ and $\hat{\theta}$.

$$D_{KL}(P_{\hat{\theta}} \| P_\theta) = \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} P_{\hat{\theta}}(s,a) \log \frac{P_{\hat{\theta}}(s,a)}{P_\theta(s,a)}$$

- The **L2 Norm** as a simpler proxy for the regret:

$$\|\theta - \hat{\theta}\|_2$$

The results are less conclusive than the ones presented by the paper, as the standard deviation from the mean is large, especially for the regret. Nevertheless, the best response generally makes the robot infer better results, closer to the true θ, than the expert demonstration.

4 Real CIRL

4.1 Motivation

After introducing the CIRL framework, the paper makes a number of strong assumptions (many being unstated) in order to apply it to its robotics application. This approach results in a loss of generality which does not fully tap into the whole potential of the framework introduced. Indeed, the case developed in the above part has more to do with optimal teaching than a collaborative game (which is still interesting but more restrictive).

We therefore wanted to work in a more general setting and use an algorithm specifically devised for the structure of CIRL problems, and which can solve them without loss of generality. This approach has enabled us to avoid using approximation tricks with functions developed for other applications (as above with the specific Maximum-Entropy IRL algorithm).

In this part, we develop an algorithm to compute exact solutions of the CIRL, that is more efficient than default methods.

4.2 Solving POMDP

There exist several standard algorithms to fully solve POMDPs. One of the exact methods is the *Witness Algorithm*:

Algorithm 1: Witness Algorithm

1. **Initialize:** $\Gamma \leftarrow$ set of height-1 plans (i.e., actions)
   $W \leftarrow \emptyset$ (set of witnesses)
2. **for** t in $\{1, 2, \ldots, T\}$ **do**
   (a) $\Gamma' \leftarrow \Gamma$
   (b) $\Gamma \leftarrow$ set plans made of an action and map from observations to plans in Γ'
   (c) **for** $\sigma = (a^R, v) \in \Gamma$ **do**
      i. **for** $s \in S$ **do**
         A. **for** $a^H \in A^H$ **do**
         B. Compute:

$$\alpha_{v(a^H)}(s) = R(s) + \gamma \sum_{s'} \sum_{a^H} P(s', a^H \mid s, \sigma) \alpha_{v(a^H)}(s')$$

         C. **if** $\alpha_{v(a^H)}$ cannot be dominated by any element of W **then**
         D. $W \leftarrow W \cup \{\alpha_{v(a^H)}\}$
   (d) $\Gamma \leftarrow \text{Prune}(\Gamma, W)$
3. **Return** Γ

The *Witness Algorithm* constructs optimal conditional plans for the POMDP by iteratively refining the set of candidate plans.

Conditional Plans A conditional plan $\sigma = (a, v)$ consists of a robot action a^R and a mapping v that assigns observations to further plans. This recursive structure allows adaptive decision-making based on observations (a plan can produce a policy).

Alpha-Vectors Each plan σ is associated with an alpha-vector α, representing the value function over the state space S. For a belief state b, the value function is:

$$V(b) = \max_{\alpha \in \Gamma} \sum_{s \in S} b(s)\alpha(s).$$

Witness Set W : W is the set of non-dominated alpha-vectors. It helps prune suboptimal plans by ensuring that only the best plans remain in Γ (a plan is suboptimal if another one has a superior value for all beliefs).

Pruning and Bellman Update - *Pruning* removes plans with alpha-vectors dominated by others in W. - The computation of alpha-vectors resembles a Bellman update, recursively calculating values based on immediate rewards and future values.

4.3 Link to CIRL

As seen above, a generic POMDP is expressed as : $M_{POMDP} = \langle S, \mathcal{A}, Z_{observation}, O(\cdot \mid \cdot, \cdot), T(\cdot \mid \cdot, \cdot), \{\Theta, R(\cdot, \cdot; \cdot)\}, P_0(\cdot, \cdot), \gamma, h \rangle$ and a CIRL can be reformulated as :

$$\mathcal{M}' = \Big\{ (X \times \Theta),\ (A^{H^\Theta} \times A^R),\ (X \times A^H),\ O,\ \Theta,\ R(\cdot \mid \cdot),\ T',\ \gamma,\ P_0$$

The complexity of the witness algorithm on a generic POMDP is :

$$O\left(t \times |S|^3 \times |A|^2 \times |Z_{obs}|^{t+1}\right)$$

Hence, on our CIRL, the complexity is :

$$O\left(t \times |X|^3 \times |\Theta|^3 \times |A^H|^{2|\Theta|} \times |A^R|^2 \times |X|^{t+1} \times |A^H|^{t+1}\right)$$

Which is prohibitively expensive, even on small problems as the term $|A^H|^{2|\Theta|}$ is extremely costly: Θ is often a very large space. This is probably why the original article chose not to fully exploit the structure of the problem to find solutions.

4.4 A more efficient formulation

The tremendous cost of computation comes from the fact that the action space considered is extremely large : for each action, we have to consider every combination of a robot action and of every possible response plan of the human to a given θ. However, it is possible to reduce the complexity by this term.

Indeed, in one of the proofs in the original article, the authors highlight that the human knows the policy of the robot, and can anticipate its response to any action. That is what we are going to exploit here: instead of considering every possible human action we will only take the optimal one to maximize future reward (this can be computed thanks to the recursive nature of the algorithm). This approach was inspired by an article on a similar subject [2].

For this we slightly reformulate the framework : The action space is reduced to $\mathcal{A}^{\mathcal{H}}$, (the actions of the human are "automated", as we will see), and the observation is the human action (for the robot). We need to introduce the Q, which corresponds to the expected value starting at state s, taking action a^H and a^R, given the mapping from observations to plan v of the robot, and for reward θ :

$$Q(x, a^H, a^R, v, \theta) = \sum_{x'} T(x, a^H, a^R, x')\alpha_{v(a^H)}(x', \theta)$$

At each step, the human takes the optimal action a^{H*} :

$$a^{H*} = \arg\max_{a^H} Q(x, a^H, a^R, v, \theta)$$

This way the update of the *alpha* vector can be made with the following formula :

$$\alpha_\sigma(x, \theta) = R(x, \theta) + \gamma \sum_{x'} T(x, a^{H*}, a^R, x')\alpha_{v(a^{H*})}(x', \theta)$$

with $\sigma = (a^R, v)$

This double recursive structure converges as the length of the conditional plan decreases at each call. This enables us to adapt the witness algorithm.

Algorithm 2: Witness Algorithm with Q-function and Optimal Human Action

1. **Initialize:** $\Gamma \leftarrow$ set of height-1 plans; $W \leftarrow \emptyset$
2. **for** $t \in \{1, 2, \ldots, T\}$ **do**

   $\Gamma' \leftarrow \Gamma$; $\Gamma \leftarrow$ set of plans with a^R and mapping v from observations to plans in Γ'
3. **for each** $\sigma = (a^R, v) \in \Gamma$ **do**

   for $x \in X$ and $\theta \in \Theta$ **do**

   Compute:

   $$a^{H*} = \arg\max_{a^H} Q(x, a^H, a^R, v, \theta)$$

   Update:

   $$\alpha_\sigma(x, \theta) = R(x, \theta) + \gamma \sum_{x'} T(x, a^{H*}, a^R, x') \alpha_{v(a^{H*})}(x', \theta)$$

   if α_σ not dominated by any $\alpha \in W$ **then** $W \leftarrow W \cup \{\alpha_\sigma\}$
4. $\Gamma \leftarrow \text{Prune}(\Gamma, W)$
5. **Return** Γ

4.5 Our implementation

We coded from scratch a general solver, which outputs optimal policies if a valid CIRL problem is given as an input, as well as an initial belief state.

4.6 Application : collaboration gridSpace

To apply this new approach, we chose to work on a step by step collaborative version of the problem developed in the article. The "robot" and the "human" are 2 players which can control a moving object. They both choose a decision at each turn, and the robot updates its belief based on the human's actions. Here, the feature centers (which correspond to gaussians) can have either, positive, negative or zero value ($\theta_i \in \{-1, 0, 1\}$). In figure 4, we see that the human indicates that the high value spot is up, and the robot acts coherently as this single action is sufficient for it to understand that the higher spot has a positive value (its belief is updated in this way).

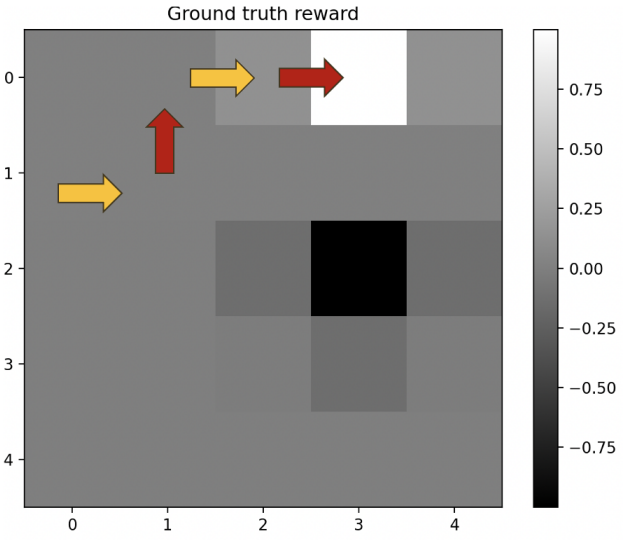

Figure 4: Example of a trajectory on a small grid (yellow corresponds to robot actions, while red corresponds to human actions

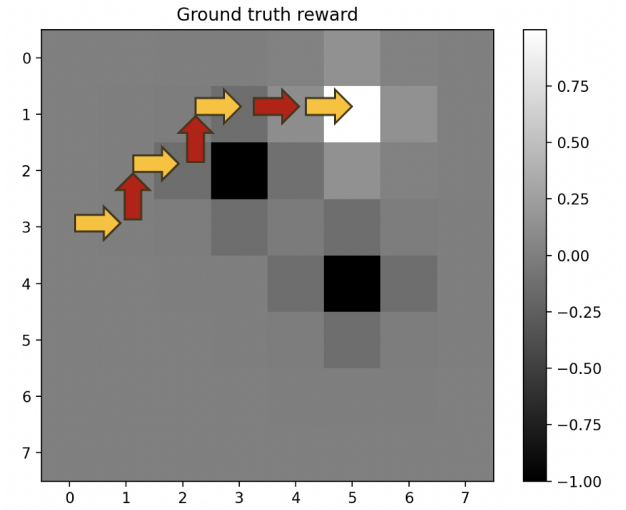

Figure 5: Example of a trajectory on a larger grid (yellow corresponds to robot actions, while red corresponds to human actions

The example in figure 5 illustrates that the robot does not have perfect information : on its second step, it chooses to go right (which is suboptimal) because the human has not yet made any action that indicates that the spot in (3,2) has a negative value (the human would have gone up if it did). The robot "understands" this (which means it updates its belief accordingly) afterwards, as the moves towards the spot in (5,1) indicate.

This example, and its subtlelties illustrate the class of complex problems that can be solved with the CIRL approach.

5 Conclusions

The Cooperative Inverse Reinforcement Learning (CIRL) framework, introduced in the foundational paper, offers an innovative approach to solving the alignment problem in human-robot collaboration. By reformulating CIRL as a Partially Observable Markov Decision Process (POMDP), the original work establishes a solid theoretical foundation and demonstrates its potential in controlled environments.

We implemented a turn-based Apprenticeship CIRL (ACIRL) framework in a 10x10 gridworld environment, addressing gaps in the original methodology with detailed explanations of key processes, such as trajectory generation, feature matching, and maximum entropy IRL. Additionally, we implemented a general CIRL algorithm, which is more precise and comprehensive than ACIRL.

A major limitation of the original framework is its computational complexity. To address this, we drew inspiration from recent research [2]. By automating the human's optimal actions and leveraging recursive updates, we reduced the action space size and computational cost, making policy computation more efficient—though it remains computationally expensive.

Our work focused on fully leveraging the formalism introduced in the article. Future directions could include exploring scenarios where the "human" has only partial information or where multiple robots are involved.

References

[1] Maximilian Luz. Maximum entropy inverse reinforcement learning, 08 2022. Accessed: 2024-12-14.

[2] Malayandi Palaniappan, Dhruv Malik, Dylan Hadfield-Menell, Anca Dragan, and Stuart Russell. Efficient cooperative inverse reinforcement learning. *ArXiv preprint*, 2023. Accessed: 2024-12-14.

[3] Richard D. Smallwood and Edward J. Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088, 1973.

[4] Brian Ziebart, Andrew Maas, and J. Bagnell. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 1433–1438, 01 2008.